Московский государственный технический университет
им. Н. Э. Баумана

УТВЕРЖДАЮ:

Большаков С.А.

"____"_____2021г.

Курсовая работа по курсу «Системное программирование»
«Резидентная программа (TSR)»

Исходный текст программного продукта
(вид документа)

писчая бумага
(вид носителя)

40
(количество листов)

ИСПОЛНИТЕЛИ:

студент группы ИУ5-41Б _____
Анцифров Н. С.
"___"_____2021 г.

Москва – 2021

```
Turbo Assembler        Version 3.1        05/30/21 15:54:02      Page 1
tsr.asm
    1                                  ; tsr.asm
    2                                  ;
    3                                  ; Сборка:
    4                                  ; tasm.exe   /l tsr.asm
    5                                  ; tlink /t      /x tsr.obj
    6                                  ; Автор:
    7                                  ; МГТУ им.          Н.Э. Баумана, ИУ5-41Б, 2021 год
    8                                  ; Анцифров          Н.С.
    9                                  ;
   10
   11  0000                            code segment    'code'
   12                                          assume  CS:code, DS:code
   13                                          org     100h
   14  0100                                    _start:
   15
   16  0100  E9 06F6                                   jmp _initTSR ; начало программы
   17
   18                                          ; Данные
   19  0103  A0 A1 A2 A3 A4 A5   F1+           ignoredChars                    DB
'абвгдеёжзийклмнопрстуфхцчшщъыьэюя'    +
   20        A6 A7 A8 A9 AA AB   AC+  ; что менять
   21        AD AE AF E0 E1 E2   E3+
   22        E4 E5 E6 E7 E8 E9   EA+
   23        EB EC ED EE EF
   24  0124  66 2C 64 75 6C 74   60+           replaceWith                     DB
'f,dult`;pbqrkvyjghcnea[wxio]sm|.z'    +
   25        3B 70 62 71 72 6B   76+  ; на что менять
   26        79 6A 67 68 63 6E   65+
   27        61 5B 77 78 69 6F   5D+
   28        73 6D 7C 2E 7A
   29        =0042                           ignoredLength                   equ     $-
ignoredChars                    +
   30                                  ; длина строки ignoredChars
   31  0145  00                               ignoreEnabled                   DB      0
                                   +
   32                                  ; работа функции игнорирования ввода
   33  0146  54 3B 50 42 52                   translateFrom                   DB      'T;PBR'
                                   +
   34                                  ; что русифицировать
   35  014B  85 86 87 88 8A                           translateTo             DB      'ЕЖЗИК'
                                   +
   36                                  ; на что русифицировать
   37        =0005                           translateLength                 equ     $-
translateTo                     +
   38                                  ; длина строки trasnlateFrom
   39  0150  00                               translateEnabled                        DB      0
                                   +
   40                                  ; работа функции перевода
   41  0151  00                               signaturePrintingEnabled                DB      0
                                   +
```

```
                        ; работа функции вывода информацииоб авторе
0152 00                         cursiveEnabled                  DB      0
                        +
                        ; работа перевода символа в курсив

0153 00                         cursiveSymbol                   DB 00000000b
0154 00                                                         DB 00000000b
0155 00                                                         DB 00000000b
0156 1F                                                         DB 00011111b
0157 18                                                         DB 00011000b
0158 30                                                         DB 00110000b
0159 30                                                         DB 00110000b
015A 6C                                                         DB 01101100b
015B 63                                                         DB 01100011b
015C CC                                                         DB 11001100b
015D F0                                                         DB 11110000b
015E 00                                                         DB 00000000b
```

```
   58  015F  00                                                    DB 00000000b
   59  0160  00                                                    DB 00000000b
   60  0161  00                                                    DB 00000000b
   61  0162  00                                                    DB 00000000b
   62
   63  0163  81                    charToCursiveIndex
         DB 'Б'                    +
   64                             ; символ для курсива
   65  0164  10*(FF)               savedSymbol                     DB 16
dup(0FFh)                        +
   66                             ; переменная для хранения старого символа
   67
   68     =00FF                    true                    equ    0FFh
                          +
   69                             ; константа истинности
   70  0174  ????                  old_int9hOffset                 DW    ?
                                  +
   71                             ; адрес старого обработчика int 9h
   72  0176  ????                  old_int9hSegment                DW    ?
                                  +
   73                             ; сегмент старого обработчика int 9h
   74  0178  ????                  old_int1ChOffset                DW    ?
                                  +
   75                             ; адрес старого обработчика int 1Ch
   76  017A  ????                  old_int1ChSegment
DW       ?                        +
   77                             ; сегмент старого обработчика int 1Ch
   78  017C  ????                  old_int2FhOffset                DW    ?
                                  +
   79                             ; адрес старого обработчика int 2Fh
   80  017E  ????                  old_int2FhSegment
DW       ?                        +
   81                             ; сегмент старого обработчика int 2Fh
   82
   83  0180  00                    notLoadTSR                      DB    0
                                  +
   84                             ; 1 - не загружать TSR
   85  0181  0000                  counter                 DW    0
   86     =0005                    printDelay              equ   5
                                  +
   87                             ; задержка перед выводом подписи в   секундах
   88
   89  0183  B3 80 AD E6 A8 E4    E0+       signatureLine1                DB
179, 'Анцифров Никита Сергеевич   +
   90        AE A2 20 8D A8 AA     A8+ ', 179
   91        E2 A0 20 91 A5 E0     A3+
   92        A5 A5 A2 A8 E7 20     20+
```

```
    93          20 20 20 20 20 2020+
    94          20 20 20 20 20 2020+
    95          20 20 20 20 20 2020+
    96          20 20 20 20 20 2020+
    97          20 20 20 20 20 2020+
    98          20 20 20 20 20 2020+
    99          20 20 20 B3
   100          =004A                       Line1_length                    equ     $-
signatureLine1
   101  01CD    B3 88 93 35 2D 34     31+       signatureLine2                DB
179, 'ИУ5-41Б                          +
   102          81 20 20 20 20 2020+  ',  179
   103          20 20 20 20 20 2020+
   104          20 20 20 20 20 2020+
   105          20 20 20 20 20 2020+
   106          20 20 20 20 20 2020+
   107          20 20 09 09 09 2020+
   108          20 20 20 20 20 20B3
   109          =0038                       Line2_length                    equ     $-
signatureLine2
   110  0205    B3 82 A0 E0 A8 A0     AD+       signatureLine3                DB
179, 'Вариант 4                        +
   111          E2 20 34 20 20 20     20+  ',  179
   112          20 20 20 20 20 2020+
   113          20 20 20 20 20 2020+
   114          20 20 20 20 20 2020+
```

```
   115      20 20 20 20 20 2020+
   116      09 09 09 20 20 2020+
   117      20 20 20 20 B3
   118      =0036                    Line3_length                    equ    $-
signatureLine3
   119 023B 20 20 91 AF E0 A0   A2+      helpMsg              DB '  Справка:', 10, 13
   120      AA A0 3A 0A 0D
   121 0247 20 20 2F 3F 09 09   20+                    DB '  /?                -
       вывод справки',      10, 13
   122      2D 20 A2 EB A2 AE   A4+
   123      20 E1 AF E0 A0 A2   AA+
   124      A8 0A 0D
   125 025F 20 20 43 74 72 6C   2B+                    DB '  Ctrl+U/Ctrl+u  -
выгрузка резидента из памяти', 10, 13
   126      55 2F 43 74 72 6C   2B+
   127      75 20 20 2D 20 A2   EB+
   128      A3 E0 E3 A7 AA A0   20+
   129      E0 A5 A7 A8 A4 A5   AD+
   130      E2 A0 20 A8 A7 20   AF+
   131      A0 AC EF E2 A8 0A   0D
   132 0290 20 20 46 32 09 09   20+                    DB '  F2                -
       вывод ФИО и группы через 5 секунд вверху    +
   133      2D 20 A2 EB A2 AE   A4+  экрана', 10, 13
   134      20 94 88 8E 20 A8   20+
   135      A3 E0 E3 AF AF EB   20+
   136      E7 A5 E0 A5 A7 20   35+
   137      20 E1 A5 AA E3 AD   A4+
   138      20 A2 A2 A5 E0 E5   E3+
   139      20 ED AA E0 A0 AD   A0+
   140      0A 0D
   141 02CA 20 20 46 33 09 09   20+                    DB '  F3                -
       включение и отключение курсивного вывода    +
   142      2D 20 A2 AA AB EE   E7+  русского символа Б', 10, 13
   143      A5 AD A8 A5 20 A8   20+
   144      AE E2 AA AB EE E7   A5+
   145      AD A8 A5 20 AA E3   E0+
   146      E1 A8 A2 AD AE A3   AE+
   147      20 A2 EB A2 AE A4   A0+
   148      20 E0 E3 E1 E1 AA   AE+
   149      A3 AE 20 E1 A8 AC   A2+
   150      AE AB A0 20 81 0A   0D
   151 0310 20 20 46 34 09 09   20+                    DB '  F4                -
       включение и отключение русификации       +
   152      2D 20 A2 AA AB EE   E7+  подмножества букв ЕЖЗИК', 10, 13
   153      A5 AD A8 A5 20 A8   20+
   154      AE E2 AA AB EE E7   A5+
   155      AD A8 A5 20 E0 E3   E1+
```

```
   156       A8 E4 A8 AA A0 E6      A8+
   157       A8 20 AF AE A4 AC      AD+
   158       AE A6 A5 E1 E2 A2      A0+
   159       20 A1 E3 AA A2 20      85+
   160       86 87 88 8A 0A 0D
   161  0355  20 20 46 35 09 09      20+                    DB '  F5                 -
         замена ввода русских букв на латинские', 10,+
   162       2D 20 A7 A0 AC A5      AD+  13
   163       A0 20 A2 A2 AE A4      A0+
   164       20 E0 E3 E1 E1 AA      A8+
   165       E5 20 A1 E3 AA A2      20+
   166       AD A0 20 AB A0 E2      A8+
   167       AD E1 AA A8 A5 0A      0D
   168
   169       =014B                      helpMsg_length                    equ    $-helpMsg
   170  0386  8E E8 A8 A1 AA A0   20+          errorParamMsg                      DB
'Ошибка параметровКС', 10, 13
   171       AF A0 E0 A0 AC A5      E2+
```

```
   172       E0 AE A2 20 8A 91       0A+
   173       0D
   174       =0016                       errorParamMsg_length            equ    $-
errorParamMsg
   175
   176  039C  DA 48*(C4) BF                  tableTop
DB         218, Line1_length-2 dup (196), +
   177                                191
   178       =004A                       tableTop_length                      equ    $-
tableTop
   179  03E6  C0 48*(C4) D9              tableBottom                    DB     192,
Line1_length-2 dup (196), 217
   180       =004A                       tableBottom_length
equ    $-tableBottom
   181
   182                                   ; сообщения
   183  0430  93 E1 AF A5 E8 AD   A0+       installedMsg                       DB
'Успешная загрузка резидента$'
   184       EF 20 A7 A0 A3 E0       E3+
   185       A7 AA A0 20 E0 A5       A7+
   186       A8 A4 A5 AD E2 A0       24
   187  044C  90 A5 A7 A8 A4 A5   AD+       alreadyInstalledMsg
DB      'Резидент уже был загружен$'
   188       E2 20 E3 A6 A5 20       A1+
   189       EB AB 20 A7 A0 A3       E0+
   190       E3 A6 A5 AD 24
   191  0466  8D A5 A4 AE E1 E2   A0+       noMemMsg
            DB  'Недостаточно памяти$'
   192       E2 AE E7 AD AE 20       AF+
   193       A0 AC EF E2 A8 24
   194  047A  8E E8 A8 A1 AA A0  20+       notInstalledMsg                    DB
'Ошибка загрузки резидента$'
   195       A7 A0 A3 E0 E3 A7       AA+
   196       A8 20 E0 A5 A7 A8       A4+
   197       A5 AD E2 A0 24
   198  0494  90 A5 A7 A8 A4 A5   AD+       removedMsg
DB      'Резидент выгружен'
   199       E2 20 A2 EB A3 E0       E3+
   200       A6 A5 AD
   201       =0011                       removedMsg_length
equ  $-removedMsg
   202  04A5  8E E8 A8 A1 AA A0  20+       noRemoveMsg                        DB
'Ошибка выгрузки резидента$'
   203       A2 EB A3 E0 E3 A7       AA+
   204       A8 20 E0 A5 A7 A8       A4+
   205       A5 AD E2 A0 24
```

```
206      =001A                  noRemoveMsg_length
         equ  $-noRemoveMsg
207
208 04BF  46 32                 f2_txt                  DB    'F2'
209 04C1  46 33                 f3_txt                  DB    'F3'
210 04C3  46 34                 f4_txt                  DB    'F4'
211 04C5  46 35                 f5_txt                  DB    'F5'
212      =0002                  fx_length               equ   $-
f5_txt
213
214 04C7                        changeFx proc
215 04C7  50                            push AX
216 04C8  53                            push BX
217 04C9  51                            push CX
218 04CA  52                            push DX
219 04CB  55                            push BP
220 04CC  06                            push ES
221 04CD  33 DB                         xor BX, BX
222
223 04CF  B4 03                         mov AH, 03h
224 04D1  CD 10                         int 10h
225 04D3  52                            push DX
226
227 04D4  0E                            push CS
228 04D5  07                            pop ES
```

```
                                
04D6                            _checkF2:
04D6  BD 04BFr                          lea BP, f2_txt
04D9  B9 0002                           mov CX, fx_length
04DC  B7 00                     mov BH, 0
04DE  B6 00                     mov DH, 0
04E0  B2 4E                     mov DL, 78
04E2  B8 1301                           mov AX, 1301h
                                
04E5  80 3E 0151r FF                    cmp signaturePrintingEnabled, true
04EA  74 07                     je    _greenF2
                                
04EC                            _redF2:
04EC  B3 4F                             mov BL, 01001111b ; red
04EE  CD 10                             int 10h
04F0  EB 08 90                                  jmp _checkF3
                                
04F3                            _greenF2:
04F3  BD 04BFr                                  lea BP, f2_txt
04F6  B3 2F                             mov BL, 00101111b ; green
04F8  CD 10                             int 10h
                                
04FA                            _checkF3:
04FA  BD 04C1r                          lea BP, f3_txt
04FD  B9 0002                           mov CX, fx_length
0500  B7 00                     mov BH, 0
0502  B6 01                     mov DH, 1
0504  B2 4E                     mov DL, 78
0506  B8 1301                           mov AX, 1301h
                                
0509  80 3E 0152r FF            cmp cursiveEnabled, true
050E  74 07                     je    _greenF3
                                
0510                            _redF3:
0510  B3 4F                             mov BL, 01001111b ; red
0512  CD 10                             int 10h
0514  EB 05 90                                  jmp _checkF4
                                
0517                            _greenF3:
0517  B3 2F                             mov BL, 00101111b ; green
0519  CD 10                             int 10h
                                
051B                            _checkF4:
051B  BD 04C3r                          lea BP, f4_txt
051E  B9 0002                           mov CX, fx_length
0521  B7 00                     mov BH, 0
0523  B6 02                     mov DH, 2
```

```
276  0525  B2 4E                    mov DL, 78
277  0527  B8 1301                        mov AX, 1301h
278
279  052A  80 3E 0150r FF                 cmp translateEnabled, true
280  052F  74 07                  je    _greenF4
281
282  0531                         _redF4:
283  0531  B3 4F                          mov BL, 01001111b ; red
284  0533  CD 10                          int 10h
285  0535  EB 05 90                               jmp _checkF5
```

```
                                _greenF4:
0538                                    mov BL, 00101111b ; green
0538  B3 2F                             int 10h
053A  CD 10
                                _checkF5:
053C                                    lea BP, f5_txt
053C  BD 04C5r                          mov CX, fx_length
053F  B9 0002                   mov BH, 0
0542  B7 00                     mov DH, 3
0544  B6 03                     mov DL, 78
0546  B2 4E                             mov AX, 1301h
0548  B8 1301
                                        cmp ignoreEnabled,      true
054B  80 3E 0145r FF            je  _greenF5
0550  74 07
                                _redF5:
0552                                    mov BL, 01001111b ; red
0552  B3 4F                             int 10h
0554  CD 10                                     jmp _outFx
0556  EB 05 90
                                _greenF5:
0559                                    mov BL, 00101111b ; green
0559  B3 2F                             int 10h
055B  CD 10
                                _outFx:
055D                            pop DX
055D  5A                        mov AH, 02h
055E  B4 02                     int 10h
0560  CD 10
                                pop ES
0562  07                        pop BP
0563  5D                       pop DX
0564  5A                       pop CX
0565  59                       pop BX
0566  5B                       pop AX
0567  58                       ret
0568  C3                    changeFx endp
0569
                              ;новый        обработчик
0569                          new_int9h proc        far
0569  56                      push SI
056A  50                      push AX
056B  53                      push BX
056C  51                      push CX
056D  52                      push DX
056E  06                      push ES
```

```
   333  056F  1E                          push DS
   334                                              ; синхронизируем CS и DS
   335  0570  0E                          push CS
   336  0571  1F                          pop  DS
   337
   338  0572  B8 0040                          mov     AX, 40h         ; 40h-
сегмент,где хранятся        флаги сост-я клавиатуры,    +
   339                          кольц. буфер ввода
   340  0575  8E C0                          mov     ES, AX
   341  0577  E4 60                          in      AL, 60h         ; записываем в AL
скан-код        нажатой         клавиши
   342
```

```
343 0579 3C 16                  cmp  AL, 22  ; проверка нажатия
    U
344 057B 75 24                  jne  _test_Fx
345 057D 26: 8A 26 0017             mov  AH, ES:[17h]  ; флаги
клавиатуры
346 0582 80 E4 0F                   and  AH, 00001111b
347 0585 80 FC 04                   cmp  AH, 00000100b  ;
    проверка нажатия ctrl
348 0588 75 17                  jne  _test_Fx
349
350                             ; выгрузка
351 058A B4 FF                      mov AH, 0FFh
352 058C B0 01                      mov AL, 01h
353 058E CD 2F                      int 2Fh
354
355 0590 E4 61                      in   AL, 61h          ;
контроллер состояния клавиатуры
356 0592 0C 80                      or   AL, 10000000b  ; клавишу
нажали
357 0594 E6 61                      out  61h, AL
358 0596 24 7F                      and  AL, 01111111b  ; клавишу
отпустили
359 0598 E6 61                      out  61h, AL
360 059A B0 20                      mov  AL, 20h
361 059C E6 20                      out  20h, AL        ; отправимв
контроллер прерываний  признак  +
362                         конца прерывания
363
364 059E E9 00A4                        jmp _quit
365
366 05A1                        _test_Fx:
367 05A1 2C 3A                  sub AL, 58              ; в AL теперь
номер функциональной  клавиши
368 05A3                        _F2:
369 05A3 3C 02                      cmp AL, 2 ; F2
370 05A5 75 0A                      jne _F3
371 05A7 F6 16 0151r                not signaturePrintingEnabled
372 05AB E8 FF19                        call changeFx
373 05AE EB 2E 90                       jmp _translate_or_ignore
374 05B1                        _F3:
375 05B1 3C 03                      cmp AL, 3 ; F3
376 05B3 75 0D                      jne _F4
377 05B5 F6 16 0152r                not cursiveEnabled
378 05B9 E8 FF0B                        call changeFx
379 05BC E8 01D3                        call setCursive         ;
перевод символа в курсив  и обратно
380 05BF EB 1D 90                       jmp _translate_or_ignore
```

```
   381  05C2                                  _F4:
   382  05C2  3C 04                                   cmp AL, 4 ; F4
   383  05C4  75 0A                                   jne _F5
   384  05C6  F6 16 0150r                             not translateEnabled
   385  05CA  E8 FEFA                                         call changeFx
   386  05CD  EB 0F 90                                        jmp _translate_or_ignore
   387  05D0                                  _F5:
   388  05D0  3C 05                                   cmp AL, 5 ; F5
   389  05D2  75 0A                                   jne _translate_or_ignore
   390  05D4  F6 16 0145r                             not ignoreEnabled
   391  05D8  E8 FEEC                                         call changeFx
   392  05DB  EB 01 90                                        jmp _translate_or_ignore
   393
   394  05DE                                  _translate_or_ignore:
   395
   396  05DE  9C                              pushf
   397  05DF  2E: FF 1E       0174r                   call dword  ptr CS:[old_int9hOffset]
; вызываем    стандартный обработчик       +
   398                                 прерывания
   399  05E4  B8 0040                                 mov     AX, 40h                 ;
40h-сегмент,где хранятся    флаги     +
```

```
                                состояния клавиатуры
05E7  8E C0                     mov     ES, AX
05E9  26: 8B 1E  001C           mov     BX, ES:[1Ch]        ; адрес хвоста
05EE  4B                        dec     BX                  ; переход назад к последнему введенному символу
05EF  4B                        dec     BX
05F0  83 FB 1E                  cmp     BX, 1Eh             ; проверка предела буфера
05F3  73 03                     jae     _go
05F5  BB 003C                   mov     BX, 3Ch             ; вышли за пределы => последний символ в конце +
                                буфера

05F8                    _go:
05F8  26: 8B 17                 mov DX, ES:[BX]             ; в DX 0 введённый символ
05FB  80 3E 0145r FF            cmp ignoreEnabled, true
0600  75 21                     jne _check_translate

0602  BE 0000                   mov SI, 0
0605  B9 0042                   mov CX, ignoredLength

0608                    _check_ignored:
0608  3A 94 0103r               cmp DL, ignoredChars[SI]
060C  74 06                     je  _block
060E  46                        inc SI
060F  E2 F7                     loop _check_ignored
0611  EB 10 90                  jmp _check_translate

0614                    _block:
0614  26: 89 07                 mov ES:[BX], AX
0617  33 C0                     xor AX, AX
0619  8A 84 0124r               mov AL, replaceWith[SI]
061D  26: 89 07                 mov ES:[BX], AX             ; замена символа
0620  EB 23 90                  jmp _quit

0623                    _check_translate:
0623  80 3E 0150r FF            cmp translateEnabled, true
0628  75 1B                     jne _quit

062A  BE 0000                   mov SI, 0
062D  B9 0005                   mov CX, translateLength
0630                    _check_translate_loop:
0630  3A 94 0146r               cmp DL, translateFrom[SI]
0634  74 06                     je  _translate
```

```
441  0636  46                          inc SI
442  0637  E2 F7                    loop _check_translate_loop
443  0639  EB 0A 90                      jmp _quit
444
445  063C                          _translate:
446  063C  33 C0                         xor AX, AX
447  063E  8A 84 014Br                   mov AL, translateTo[SI]
448  0642  26: 89 07                           mov ES:[BX], AX
449
450  0645                     _quit:
451  0645  1F                        pop    DS
452  0646  07                        pop    ES
453  0647  5A                        pop DX
454  0648  59                        pop CX
455  0649  5B                        pop    BX
456  064A  58                        pop    AX
```

```
  457 064B  5E                              pop SI
  458 064C  CF                              iret
  459 064D                        new_int9h endp
  460
  461                             ; Обработчик прерывания int 1Ch
  462                             ; Вызывается каждые 55 мс
  463 064D                        new_int1Ch      proc far
  464 064D  50                          push AX
  465 064E  0E                          push CS
  466 064F  1F                              pop DS
  467
  468 0650  9C                          pushf
  469 0651  2E: FF 1E     0178r             call dword  ptr CS:[old_int1ChOffset]
  470
  471 0656  80 3E 0151r FF              cmp signaturePrintingEnabled, true
        ; если нажата управляющая       +
  472                             клавиша (в      данном случае F1)
  473 065B  75 1C                       jne _notToPrint
  474
  475 065D  83 3E 0181r 5B                      cmp counter, printDelay*1000/55 + 1
        ; если кол-во "тактов"      +
  476                             эквивалентно %printDelay% секундам
  477 0662  74 03                       je  _letsPrint
  478
  479 0664  EB 0E 90                        jmp _dontPrint
  480
  481 0667                              _letsPrint:
  482 0667  F6 16 0151r                     not signaturePrintingEnabled
  483 066B  C7 06 0181r 0000                    mov counter, 0
  484 0671  E8 0094                         call printSignature
  485
  486 0674                              _dontPrint:
  487 0674  83 06 0181r 01                  add counter, 1
  488
  489 0679                              _notToPrint:
  490
  491 0679  58                          pop AX
  492
  493 067A  CF                          iret
  494 067B                        new_int1Ch      endp
  495
  496                             ; Обработчик прерывания int 2Fh
  497                             ; Служит для:
  498                             ; 1) проверки факта присутствия TSR в памяти (при
AH=0FFh,    AL=0)
  499                             ;       будетвозвращён AH='i' в случае, если    TSR уже
      загружен
```

```
500                             ; 2) выгрузки TSR из памяти (при AH=0FFh, AL=1)
501                             ;
502  067B                       new_int2Fh          proc
503  067B  80 FC FF                             cmp     AH, 0FFh                        ;
наша функция?
504  067E  75 0B                        jne     _2Fh_std                        ; нет - на
       старый обработчик
505  0680  3C 00                        cmp     AL, 0                           ;
подфункция проверки, загружен ли            +
506                             резидент в память?
507  0682  74 0C                        je      _already_installed
508  0684  3C 01                        cmp     AL, 1                           ; подфункция
выгрузки из памяти?
509  0686  74 0B                        je      _uninstall
510  0688  EB 01 90                             jmp     _2Fh_std                        ;
нет - на         старый обработчик
511
512  068B                       _2Fh_std:
513  068B  2E: FF 2E       017Cr                jmp     dword ptr CS:[old_int2FhOffset] ;
вызов старого обработчика
```

```
514
515 0690                    _already_installed:
516 0690 B4 69                          mov     AH, 'i'                 ; вернём 'i', если
    резидент загружен в память
517 0692 CF                             iret
518
519 0693                    _uninstall:
520 0693 1E                         push    DS
521 0694 06                         push    ES
522 0695 52                         push    DX
523 0696 53                         push    BX
524
525 0697 33 DB                      xor BX, BX
526
527                                                                     ; CS = ES,  для
доступа к переменным
528 0699 0E                         push CS
529 069A 07                         pop ES
530
531 069B B8 2509                        mov     AX, 2509h
532 069E 26: 8B 16      0174r           mov DX, ES:old_int9hOffset
        ; возвращаем вектор прерывания на место
533 06A3 26: 8E 1E      0176r           mov DS, ES:old_int9hSegment
534 06A8 CD 21                      int     21h
535
536 06AA B8 251C                        mov     AX, 251Ch
537 06AD 26: 8B 16      0178r           mov DX, ES:old_int1ChOffset             ;
возвращаем вектор прерывания на место
538 06B2 26: 8E 1E      017Ar           mov DS, ES:old_int1ChSegment
539 06B7 CD 21                      int     21h
540
541 06B9 B8 252F                        mov     AX, 252Fh
542 06BC 26: 8B 16      017Cr           mov DX, ES:old_int2FhOffset             ;
возвращаем вектор прерывания на место
543 06C1 26: 8E 1E      017Er           mov DS, ES:old_int2FhSegment
544 06C6 CD 21                      int     21h
545
546 06C8 2E: 8E 06      002C            mov     ES, CS:2Ch
        ; загрузим в ES адрес окружения
547 06CD B4 49                      mov     AH, 49h                 ; выгрузимиз
памяти окружение
548 06CF CD 21                      int     21h
549 06D1 72 0B                      jc      _notRemove
550
551 06D3 0E                         push    CS
552 06D4 07                         pop     ES                      ; в ES - адрес
резидентной  программы
```

```
553  06D5  B4 49                mov     AH, 49h                         ;
выгрузим       из памяти резидент
554  06D7  CD 21                int     21h
555  06D9  72 03                jc    _notRemove
556  06DB  EB 15 90                   jmp _unloaded
557
558  06DE                  _notRemove:                                  ; не удалось
выполнить выгрузку
559  06DE  B4 03                mov AH, 03h                             ; получаем
     позицию       курсора
560  06E0  CD 10                int 10h
561  06E2  BD 04A5r                   lea BP, noRemoveMsg
562  06E5  B9 001A                    mov CX, noRemoveMsg_length
563  06E8  B3 07                mov BL, 0111b
564  06EA  B8 1301                    mov AX, 1301h
565  06ED  CD 10                int 10h
566  06EF  EB 12 90                   jmp _2Fh_exit
567
568  06F2                  _unloaded:                                   ; выгрузка
     прошла успешно
569  06F2  B4 03                mov AH, 03h                             ; получаем
     позицию       курсора
570  06F4  CD 10                int 10h
```

```
   571  06F6  BD 0494r                  lea BP, removedMsg
   572  06F9  B9 0011                   mov CX, removedMsg_length
   573  06FC  B3 07                 mov BL, 0111b
   574  06FE  B8 1301                   mov AX, 1301h
   575  0701  CD 10                 int 10h
   576
   577  0703                    _2Fh_exit:
   578  0703  5B                   pop BX
   579  0704  5A                   pop    DX
   580  0705  07                   pop    ES
   581  0706  1F                   pop    DS
   582  0707  CF                   iret
   583  0708                    new_int2Fh      endp
   584
   585                     ; Процедура вывода      подписи      (ФИО, группа)
   586                     ;
   587  0708                    printSignature proc
   588  0708  50                   push AX
   589  0709  52                   push DX
   590  070A  51                   push CX
   591  070B  53                   push BX
   592  070C  06                   push ES
   593  070D  54                   push SP
   594  070E  55                   push BP
   595  070F  56                   push SI
   596  0710  57                   push DI
   597
   598  0711  33 C0                xor AX, AX
   599  0713  33 DB                xor BX, BX
   600  0715  33 D2                xor DX, DX
   601
   602  0717  B4 03                mov AH, 03h                     ;
чтение текущей позиции курсора
   603  0719  CD 10                int 10h
   604  071B  52                   push DX                         ;
помещаем    информацию о        +
   605                     положении курсора в стек
   606
   607  071C                       _printTop:
   608  071C  B6 00                    mov DH, 0
   609  071E  B2 0F                    mov DL, 15
   610  0720  EB 01 90                      jmp _actualPrint
   611
   612  0723                       _actualPrint:
   613  0723  B4 0F                    mov AH, 0Fh                     ;
чтение текущего видеорежима
   614  0725  CD 10                    int 10h
```

```
615
616 0727  0E                          push CS
617 0728  07                          pop ES                          ;
указываем ES на CS
618
619                                                                   ;
вывод "верхушки"    таблицы
620 0729  52                          push DX
621 072A  BD 039Cr                          lea BP, tableTop
                ; помещаем          в BP       +
622                           указатель на выводимую строку
623 072D  B9 004A                           mov CX, tableTop_length
                ; в CX - длина строки
624 0730  B3 07                       mov BL, 0111b
625 0732  B8 1301                           mov AX, 1301h
          ; AH=13h -номер фамилии,          +
626                           AL=01h - курсор перемещается при выводе каждого из
      символов строки
627 0735  CD 10                       int 10h
```

```
  628 0737 5A                   pop DX
  629 0738 FE C6                inc DH
  630                                                                   ;
вывод первой линии
  631 073A 52                   push DX
  632 073B BD 0183r                     lea BP, signatureLine1
  633 073E B9 004A                      mov CX, Line1_length
  634 0741 B3 07                mov BL, 0111b
  635 0743 B8 1301                      mov AX, 1301h
  636 0746 CD 10                int 10h
  637 0748 5A                   pop DX
  638 0749 FE C6                inc DH
  639
  640
        ; вывод второй линии
  641 074B 52                   push DX
  642 074C BD 01CDr                     lea BP, signatureLine2
  643 074F B9 0038                      mov CX, Line2_length
  644 0752 B3 07                mov BL, 0111b
  645 0754 B8 1301                      mov AX, 1301h
  646 0757 CD 10                int 10h
  647 0759 5A                   pop DX
  648 075A FE C6                inc DH
  649
  650
        ; вывод третьей линии
  651 075C 52                   push DX
  652 075D BD 0205r                     lea BP, signatureLine3
  653 0760 B9 0036                      mov CX, Line3_length
  654 0763 B3 07                mov BL, 0111b
  655 0765 B8 1301                      mov AX, 1301h
  656 0768 CD 10                int 10h
  657 076A 5A                   pop DX
  658 076B FE C6                inc DH
  659
  660
        ; вывод 'низа' таблицы
  661 076D 52                   push DX
  662 076E BD 03E6r                     lea BP, tableBottom
  663 0771 B9 004A                      mov CX, tableBottom_length
  664 0774 B3 07                mov BL, 0111b
  665 0776 B8 1301                      mov AX, 1301h
  666 0779 CD 10                int 10h
  667 077B 5A                   pop DX
  668 077C FE C6                inc DH
  669
  670 077E 33 DB                xor BX, BX
```

```
671  0780  5A                    pop DX                    ;
восстанавливаем из стека      +
672                          прежнее положение курсора
673  0781  B4 02                 mov AH, 02h               ;
меняем положение   курсора      на  +
674                          первоначальное
675  0783  CD 10                 int 10h
676  0785  E8 FD3F                     call changeFx
677
678  0788  5F              pop DI
679  0789  5E              pop SI
680  078A  5D              pop BP
681  078B  5C              pop SP
682  078C  07              pop ES
683  078D  5B              pop BX
684  078E  59              pop CX
```

```
  685 078F 5A                   pop DX
  686 0790 58                   pop AX
  687
  688 0791 C3                   ret
  689 0792                  printSignature endp
  690
  691                       ; Изменение на курсив и наоборот
  692                       ;
  693 0792                  setCursive  proc
  694 0792 06                   push ES                         ; сохраняем
регистры
  695 0793 50                   push AX
  696 0794 0E                   push CS
  697 0795 07                   pop ES
  698
  699 0796 80 3E 0152r FF       cmp cursiveEnabled, true
  700 079B 75 30                jne _restoreSymbol
  701                                                   ; если флаг равен true,
выполняем замену символа на    +
  702                       курсивный вариант,
  703                                                   ;
предварительно сохраняя старый символ в          +
  704                       savedSymbol
  705
  706 079D E8 004C                  call saveFont
  707 07A0 8A 0E 0163r          mov CL, charToCursiveIndex
  708 07A4                  _shifTtable:
  709                                                   ; мы получаем в BP
      таблицу       всех символов. адрес          +
  710                       указывает на символ 0
  711                                                   ; поэтому нужно
совершить сдвиг 16*X, где X - код          +
  712                       символа
  713 07A4 83 C5 10                 add BP, 16
  714 07A7 E2 FB                loop _shiftTable
  715
  716                                                       ; при savefont
смещается регистр ES
  717                                                   ; поэтому приходится
делать такие махинации, чтобы
  718                                                       ; записать
      полученный элемент в savedSymbol
  719 07A9 1E                   push DS
  720 07AA 58                   pop AX
  721 07AB 06                   push ES
  722 07AC 1F                   pop DS
  723 07AD 50                   push AX
```

```
   724  07AE  07                  pop ES
   725  07AF  50                  push AX
   726
   727  07B0  8B F5               mov SI, BP
   728  07B2  BF 0164r                lea DI, savedSymbol
   729                                                   ; сохраняем в
переменную savedSymbol
   730                                                   ; таблицу нужного
символа
   731  07B5  B9 0010                 mov CX, 16
   732                                                   ; movsb из DS:SI в
        ES:DI
   733  07B8  F3> A4              rep movsb
   734                                                   ; исходные
        позиции        сегментов возвращены
   735  07BA  1F                  pop DS                 ; восстановление DS
   736
   737                                                           ; заменим
написание символа на курсив
   738  07BB  B9 0001                 mov CX, 1
   739  07BE  B6 00               mov DH, 0
   740  07C0  8A 16 0163r         mov DL, charToCursiveIndex
   741  07C4  BD 0153r                lea BP, cursiveSymbol
```

```
  742  07C7  E8 0015                    call changeFont
  743  07CA  EB 10 90                   jmp _exitSetCursive
  744
  745  07CD                     _restoreSymbol:
  746                                   ; если флаг равен 0, выполняем замену курсивного
символа на старый   вариант
  747  07CD  B9 0001                    mov CX, 1
  748  07D0  B6 00                      mov DH, 0
  749  07D2  8A 16 0163r                mov DL, charToCursiveIndex
  750  07D6  BD 0164r                   lea bp, savedSymbol
  751  07D9  E8 0003                    call changeFont
  752
  753  07DC                     _exitSetCursive:
  754  07DC  58                         pop AX
  755  07DD  07                         pop ES
  756  07DE  C3                         ret
  757  07DF                     setCursive  endp
  758
  759                           ; Функция смены начертания    символа
      (курсив/нормальное)
  760                           ;
  761                           ; Входные данные:
  762                           ; DL - номер символа для замены
  763                           ; CX - кол-во символов заменяемых изображений
символов (начиная с DX)
  764                           ; ES:bp = адрес таблицы
  765                           ;
  766                           ; Описание        работы процедуры:
  767                           ; Происходит вызов       int 10h (видеосервис)
  768                           ; с функцией AH = 11h (функции знакогенератора)
  769                           ; Параметр        AL = 0 сообщает, что будет заменено
изображение
  770                           ; символа для текущего шрифта
  771                           ; В случаях, когда        AL = 1 или 2, будет заменено
изображение
  772                           ; только для опредленного шрифта (8x14 и 8x8
соответственно)
  773                           ; Параметр        BH = 0Eh сообщает, что на определние
каждого изображения символа
  774                           ; расходуется по 14 байт (режим 8x14 бит как раз 14
байт)
  775                           ; Параметр        BL = 0 - блок шрифта для загрузки (от
0       до 4)
  776                           ;
  777
  778  07DF                     changeFont        proc
  779  07DF  50                         push AX
```

```
780  07E0  53                    push BX
781  07E1  B8 1100                    mov AX, 1100h
782  07E4  BB 1000                    mov BX, 1000h
783  07E7  CD 10                 int 10h
784  07E9  58                    pop AX
785  07EA  5B                    pop BX
786  07EB  C3                    ret
787  07EC                  changeFont        endp
788
789                        ; Нормальное начертание
790                        ;
791                        ; Входные данные:
792                        ; BH - тип возвращаемой символьной таблицы
793                        ; 0 - таблица из int 1fh
794                        ; 1 - таблица из int 44h
795                        ; 2-5 - таблица из 8x14, 8x8, 8x8 (top), 9x14
796                        ; 6 - 8x16
797                        ;
798                        ; Описание        работы процедуры:
```

```
                        ; Происходит вызов int 10h (видеосервис)
                        ; с функцией AH = 11h (функции знакогенератора)
                        ; Параметр AL = 30 - подфункция получения информации о EGA
                        ;
                        ; Результаты:
                        ; в ES:BP находится таблица символов (полная)
                        ; в CX находится байт на символ
                        ; в DL количество экранных строк
                        ; Происходит сдвиг регистра ES
                        ; (ES становится равным C000h)

07EC                    saveFont proc
07EC  50                        push AX
07ED  53                        push BX
07EE  B8 1130                           mov AX, 1130h
07F1  BB 0600                           mov BX, 0600h
07F4  CD 10                     int 10h
07F6  58                        pop AX
07F7  5B                        pop BX
07F8  C3                        ret
07F9                    saveFont endp


07F9                    _initTSR:                       ; старт резидента
07F9  B4 03                     mov AH, 03h
07FB  CD 10                     int 10h
07FD  52                        push DX
07FE  B4 00                     mov AH,00h              ; установка видеорежима (83h текст 80x25 16/8 CGA,EGA b800+
                        Comp,RGB,Enhanced), без очистки экрана
0800  B0 83                     mov AL,83h
0802  CD 10                     int 10h
0804  5A                        pop DX
0805  B4 02                     mov AH, 02h
0807  CD 10                     int 10h


0809  E8 008C                      call commandParamsParser
080C  B8 3509                       mov AX,3509h            ; получить в ES:BX вектор 09 прерывания
080F  CD 21                 int 21h


0811  80 3E 0180r FF            cmp notLoadTSR, true    ;       если была выведена справка, выход
0816  74 0E                     je  _exit_tmp
```

```
842
843  0818  B4 FF                    mov AH, 0FFh
844  081A  B0 00                    mov AL, 0
845  081C  CD 2F                    int 2Fh
846  081E  80 FC 69                       cmp AH, 'i'     ; проверка того, загружена
     ли уже программа
847  0821  74 62                    je    _alreadyInstalled
848
849  0823  EB 04 90                       jmp _tmp
850
851  0826                           _exit_tmp:
852  0826  EB 6E 90                            jmp _exit
853
854  0829                           _tmp:
855  0829  06                       push ES
```

```
   856 082A  A1 002C                    mov AX, DS:[2Ch]                ; psp
   857 082D  8E C0                      mov ES, AX
   858 082F  B4 49                      mov AH, 49h                     ; хватит лм памяти,
чтобы остаться      резидентом?
   859 0831  CD 21                      int 21h
   860 0833  07                         pop ES
   861 0834  72 59                      jc  _notMem                     ; если нет -
выход
   862
   863                                  ; int 09h
   864
   865 0836  2E: 89 1E     0174r        mov   word ptr CS:old_int9hOffset, BX
   866 083B  2E: 8C 06     0176r        mov   word ptr CS:old_int9hSegment, ES
   867 0840  B8 2509                    mov AX, 2509h                   ;
установим вектор    на 09
   868 0843  BA 0569r                   mov DX, offset new_int9h
         ; прерывание
   869 0846  CD 21                      int 21h
   870
   871                                  ; int 1Ch
   872 0848  B8 351C                    mov AX,351Ch                    ;
получить      в ES:BX      вектор 1C
   873 084B  CD 21                      int 21h                         ;
прерывания
   874 084D  2E: 89 1E     0178r        mov word ptr CS:old_int1ChOffset, BX
   875 0852  2E: 8C 06     017Ar        mov word ptr CS:old_int1ChSegment,  ES
   876 0857  B8 251C                    mov AX, 251Ch                   ;
установим вектор    на 1C
   877 085A  BA 064Dr                   mov DX, offset new_int1Ch
         ; прерывание
   878 085D  CD 21                      int 21h
   879
   880                                  ; int 2Fh
   881 085F  B8 352F                    mov AX,352Fh                    ;
получить      в ES:BX      вектор 1C
   882 0862  CD 21                      int 21h                         ;
прерывания
   883 0864  2E: 89 1E     017Cr        mov word ptr CS:old_int2FhOffset, BX
   884 0869  2E: 8C 06     017Er        mov word ptr CS:old_int2FhSegment,  ES
   885 086E  B8 252F                    mov AX, 252Fh                   ;
установим вектор    на 2F
   886 0871  BA 067Br                   mov DX, offset new_int2Fh
         ; прерывание
   887 0874  CD 21                      int 21h
   888
   889 0876  E8 FC4E                    call changeFx
   890 0879  BA 0430r                   mov DX, offset installedMsg
```

```
   891  087C  B4 09                    mov AH, 9
   892  087E  CD 21                    int 21h
   893  0880  BA 07F9r                       mov DX, offset _initTSR              ;
остаемся       резидентом и выходим
   894  0883  CD 27                    int 27h
   895
   896  0885                       _alreadyInstalled:
   897  0885  B4 09                    mov AH, 09h
   898  0887  BA 044Cr                       lea DX, alreadyInstalledMsg
   899  088A  CD 21                    int 21h
   900  088C  EB 08 90                       jmp _exit
   901  088F                       _notMem:                                        ; не
хватает памяти, чтобы       оставаться  +
   902                            резидентом
   903  088F  BA 0466r                     mov DX, offset       noMemMsg
   904  0892  B4 09                    mov AH, 9
   905  0894  CD 21                    int 21h
   906  0896                       _exit:                                          ; выход
   907  0896  CD 20                    int 20h
   908
   909                            ; Проверкапараметров КС;
   910                            ;
   911  0898                       commandParamsParser proc
   912  0898  0E                       push CS
```

```
   913  0899  07                  pop ES
   914  089A  C6 06 0180r 00               mov notLoadTSR, 0
   915
   916  089F  BE 0080                      mov SI, 80h                    ;
смещение      параметров КС
   917  08A2  AC                  lodsb                                  ; кол-во
символов.
   918  08A3  0A C0               or   AL, AL
   919  08A5  74 3A               jz   _exitHelp
   920
   921  08A7                      _nextChar:
   922
   923  08A7  46                  inc SI
   924
   925  08A8  80 3C 0D                     cmp [SI], BYTE ptr         13
   926  08AB  74 34               je   _exitHelp
   927
   928
   929  08AD  AD                           lodsw                          ;
получаем      два символа
   930  08AE  3D 3F2F                            cmp AX, '?/'
   931  08B1  74 03                        je   _question
   932
   933                                    ;cmp AH, '/'
   934                                    ;je _errorParam
   935
   936  08B3  EB 2C 90                           jmp _exitHelp
   937
   938  08B6                      _question:
   939  08B6  B4 03                        mov AH,03
   940  08B8  CD 10                        int 10h
   941  08BA  BD 023Br                           lea BP, helpMsg
   942  08BD  B9 014B                            mov CX, helpMsg_length
   943  08C0  B3 07                        mov BL, 0111b
   944  08C2  B8 1301                            mov AX, 1301h
   945  08C5  CD 10                        int 10h
   946  08C7  F6 16 0180r                  not notLoadTSR
   947  08CB  EB DA                        jmp _nextChar
   948
   949  08CD  EB 12 90                     jmp _exitHelp
   950
   951  08D0                      _errorParam:
   952  08D0  B4 03                              mov AH,03
   953  08D2  CD 10                              int 10h
   954  08D4  BD 0386r                                 lea BP, CS:errorParamMsg
   955  08D7  B9 0016                                  mov CX,
errorParamMsg_length
```

```
956  08DA  B3 07                                mov BL, 0111b
957  08DC  B8 1301                                    mov AX, 1301h
958  08DF  CD 10                                int 10h
959  08E1                              _exitHelp:
960  08E1  C3                          ret
961  08E2                        commandParamsParser endp
962
963  08E2                        code ends
964                              end _start
```

Turbo Assembler Version 3.1 05/30/21 15:54:02 Page 18
Symbol Table

Symbol Name	Type	Value
??DATE	Text	"05/30/21"
??FILENAME	Text	"tsr "
??TIME	Text	"15:54:01"
??VERSION	Number	030A
@CPU	Text	0101H
@CURSEG	Text	CODE
@FILENAME	Text	TSR
@WORDSIZE	Text	2
ALREADYINSTALLEDMSG	Byte	CODE:044C
CHANGEFONT	Near	CODE:07DF
CHANGEFX	Near	CODE:04C7
CHARTOCURSIVEINDEX	Byte	CODE:0163
COMMANDPARAMSPARSER	Near	CODE:0898
COUNTER	Word	CODE:0181
CURSIVEENABLED	Byte	CODE:0152
CURSIVESYMBOL	Byte	CODE:0153
ERRORPARAMMSG	Byte	CODE:0386
ERRORPARAMMSG_LENGTH	Number	0016
F2_TXT	Byte	CODE:04BF
F3_TXT	Byte	CODE:04C1
F4_TXT	Byte	CODE:04C3
F5_TXT	Byte	CODE:04C5
FX_LENGTH	Number	0002
HELPMSG	Byte	CODE:023B
HELPMSG_LENGTH	Number	014B
IGNOREDCHARS	Byte	CODE:0103
IGNOREDLENGTH	Number	0042
IGNOREENABLED	Byte	CODE:0145
INSTALLEDMSG	Byte	CODE:0430
LINE1_LENGTH	Number	004A
LINE2_LENGTH	Number	0038
LINE3_LENGTH	Number	0036
NEW_INT1CH	Far	CODE:064D
NEW_INT2FH	Near	CODE:067B
NEW_INT9H	Far	CODE:0569
NOMEMMSG	Byte	CODE:0466
NOREMOVEMSG	Byte	CODE:04A5
NOREMOVEMSG_LENGTH	Number	001A
NOTINSTALLEDMSG	Byte	CODE:047A
NOTLOADTSR	Byte	CODE:0180
OLD_INT1CHOFFSET	Word	CODE:0178
OLD_INT1CHSEGMENT	Word	CODE:017A
OLD_INT2FHOFFSET	Word	CODE:017C
OLD_INT2FHSEGMENT	Word	CODE:017E

```
OLD_INT9HOFFSET                 Word  CODE:0174
OLD_INT9HSEGMENT                Word  CODE:0176
PRINTDELAY                      Number 0005
PRINTSIGNATURE                  Near  CODE:0708
REMOVEDMSG                      Byte  CODE:0494
REMOVEDMSG_LENGTH               Number 0011
REPLACEWITH                     Byte  CODE:0124
SAVEDSYMBOL                     Byte  CODE:0164
SAVEFONT                  Near  CODE:07EC
SETCURSIVE                      Near  CODE:0792
```

Turbo Assembler Version 3.1 05/30/21 15:54:02 Page 19
Symbol Table

SIGNATURELINE1 Byte CODE:0183
SIGNATURELINE2 Byte CODE:01CD
SIGNATURELINE3 Byte CODE:0205
SIGNATUREPRINTINGENABLED Byte CODE:0151
TABLEBOTTOM Byte CODE:03E6
TABLEBOTTOM_LENGTH Number 004A
TABLETOP Byte CODE:039C
TABLETOP_LENGTH Number 004A
TRANSLATEENABLED Byte CODE:0150
TRANSLATEFROM Byte CODE:0146
TRANSLATELENGTH Number 0005
TRANSLATETO Byte CODE:014B
TRUE Number 00FF
_2FH_EXIT Near CODE:0703
_2FH_STD Near CODE:068B
_ACTUALPRINT Near CODE:0723
_ALREADYINSTALLED Near CODE:0885
_ALREADY_INSTALLED Near CODE:0690
_BLOCK Near CODE:0614
_CHECKF2 Near CODE:04D6
_CHECKF3 Near CODE:04FA
_CHECKF4 Near CODE:051B
_CHECKF5 Near CODE:053C
_CHECK_IGNORED Near CODE:0608
_CHECK_TRANSLATE Near CODE:0623
_CHECK_TRANSLATE_LOOP Near CODE:0630
_DONTPRINT Near CODE:0674
_ERRORPARAM Near CODE:08D0
_EXIT Near CODE:0896
_EXITHELP Near CODE:08E1
_EXITSETCURSIVE Near CODE:07DC
_EXIT_TMP Near CODE:0826
_F2 Near CODE:05A3
_F3 Near CODE:05B1
_F4 Near CODE:05C2
_F5 Near CODE:05D0
_GO Near CODE:05F8
_GREENF2 Near CODE:04F3
_GREENF3 Near CODE:0517
_GREENF4 Near CODE:0538
_GREENF5 Near CODE:0559
_INITTSR Near CODE:07F9
_LETSPRINT Near CODE:0667
_NEXTCHAR Near CODE:08A7
_NOTMEM Near CODE:088F
_NOTREMOVE Near CODE:06DE
_NOTTOPRINT Near CODE:0679

```
_OUTFX                  Near   CODE:055D
_PRINTTOP               Near   CODE:071C
_QUESTION               Near   CODE:08B6
_QUIT                   Near   CODE:0645
_REDF2                  Near   CODE:04EC
_REDF3                  Near   CODE:0510
_REDF4                  Near   CODE:0531
_REDF5                  Near   CODE:0552
_RESTORESYMBOL          Near   CODE:07CD
_SHIFTTABLE             Near   CODE:07A4
```

Turbo Assembler Version 3.1 05/30/21 15:54:02 Page 20
Symbol Table

_START		Near	CODE:0100
_TEST_FX		Near	CODE:05A1
_TMP		Near	CODE:0829
_TRANSLATE		Near	CODE:063C
_TRANSLATE_OR_IGNORE		Near	CODE:05DE
_UNINSTALL		Near	CODE:0693
_UNLOADED		Near	CODE:06F2

Groups & Segments	Bit	Size	Align	Combine	Class
CODE	16	08E2	Para	none	CODE